Asynchronous Stimulation for Cochlear Implants

A Thesis

Presented in Partial Fulfillment of the Requirements for
the Degree Bachelor of Science with Honors Research Distinction in
Electrical and Computer Engineering

By

Tom Zajdel

* * * * *

Electrical and Computer Engineering

The Ohio State University

2012

Examination Committee:

Prof. Steven Bibyk, Co-Adviser

Prof. Bomjun Kwon, Co-Adviser

Prof. Hooshang Hemami

ABSTRACT

This thesis contributes the first psychoacoustic tests of Asynchronous Interleaved Sampling (AIS) performed with commercially available implants. AIS was originally proposed as an alternative to Continuous Interleaved Sampling (CIS) to demonstrate the benefit of asynchronous stimulation on conveying phase information. AIS automatically adjusts each channel's stimulation rate based on its relative temporal importance, mimicking the behavior of neurons. However, the algorithm was never tested with actual CI patients, so its claim of improving phase perception has stood unsubstantiated. The software design and implementation of AIS lays the groundwork for the first qualitative tests of the method's effect on phase discrimination. CI users participated in Schroeder phase discrimination (SPD) experiments to evaluate AIS's presentation of phase and temporal information. AIS performs comparably to CIS in SPD and in some cases introduces a new intelligible temporal cue. The results suggest that consideration of other factors such as electrode spacing and parameter adjustment may have the potential to improve the performance of asynchronous stimulation algorithms in CI devices. Asynchronous stimulation may also be applied to other technology, and represents a subject with rich synergy between engineering and biology.

This is dedicated to my family, teachers, and mentors who have supported and guided me throughout the meandering explorations of undergraduate education.

ACKNOWLEDGMENTS

First of all, I thank Bomjun Kwon, my speech and hearing science adviser, for granting me the opportunity to contribute to cochlear implant research. He generously donated his time to orient me in the field of audiology, and our long discussions have shown me many interesting facets of audition and sound. His access to Cochlear Corporation implant test equipment and software was invaluable.

I thank Steven Bibyk, my engineering adviser, for making this collaboration possible and offering advice on which aspects of this inquiry are interesting from the circuit designer's perspective.

I thank Hooshang Hemami for taking the time to serve on my thesis committee. His encouragement and enthusiasm for all things biological and philosophical helped me consider things from a different viewpoint.

I thank the members of the Auditory Implant Speech Perception Laboratory: Jae Park, Cassie Wilhelm, and Trevor Perry, all of whom I collaborated with on the project. Jae taught me the basics of the Nucleus MATLAB Toolbox software package and also how to set up the experimental hardware. Trevor and Cassie graciously helped me arrange testing with human subjects while balancing their myriad duties as graduate students.

Most importantly, this research would have been impossible without the brave implant users who volunteered to be exposed to these novel stimulation algorithms. I am

grateful for their continued patience, cooperation, humor, and enthusiasm throughout the entire process.

This research was funded by an Undergraduate Research Scholarship granted by the Ohio State University College of Engineering.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement & Motivation

For 286 million people in the world, the sense of hearing is completely nonexistent. These profoundly deaf had to resort to sign language and lip reading to communicate with others until the advent of the cochlear implant (CI) in the 1970s. CIs restore some sense of hearing by directly stimulating the auditory nerve with a surgically implanted electrode array. To date, over 200,000 patients worldwide have been implanted [1]. Although many patients have had success understanding speech in quiet environments, their ability degrades quickly with the addition of background noise. In addition to poor noise performance, CI users have difficulty interpreting finer sounds such as speech intonation and the subtler tonal variations of music [2, 3, 4]. These sound cues require the decoding of quick temporal variations to be understood, and it has been theorized that information about phase is just as critical to understanding these cues as is information about their frequency content [5].

The present industry standard for CI sound processing is based on Continuous Interleaved Sampling (CIS), which uses interleaved pulse trains of electrical current to stimulate the auditory nerve. Because the pulses are interleaved, only one electrode is activated at once. In this way severe electrode interaction is avoided, preventing

degradation of the implant's spectral resolution. The problem with CIS is that the amount of time between pulses, the inter-spike interval, is held constant while natural hearing relies on variations in the inter-spike interval to convey a sound's temporal information. As a result, electrical stimuli produced using CIS are not nearly as adaptive to rapid temporal variations as natural neural responses are.

Asynchronous Interleaved Sampling (AIS) is a proposed alternative to CIS that mimics the integrate-and-fire response neurons use to adapt their firing rates to the input sound characteristics. When a particular spectral channel changes rapidly, the channel's firing rate is adjusted naturally as a result of its integration response. Waveform reconstructions formed from AIS pulse patterns correlate more strongly to their source sound signal's phase than their CIS counterparts, suggesting that AIS preserves phase better than CIS. Psychoacoustic tests have been performed on normal hearing (NH) subjects to support this conclusion, showing that NH listeners could more easily identify segments of music and other complex sound signals encoded by AIS than those encoded by CIS [1]. However, these tests have never been performed on actual CI users.

If AIS indeed presents a better stimulus to a CI user than does CIS, a higher quality of hearing can be achieved. With better phase presentation, CI users could become better at understanding speech intonation or music and Mandarin Chinese speakers would have an easier time understanding the fine tonal nuances that are important for that language's meaning. Some CI patients who have been postlingually deafened are already successful at understanding speech, but children born deaf have not had the opportunity to learn to speak and thus their auditory system develops with poorer

quality CIS signals. These prelingually deaf patients have the most to gain from an improvement in the electrical stimulus.

1.2 Contribution

The primary contribution of this thesis is the first test of AIS's ability to deliver phase information to CI patients through commercially available implants. An engineering design process has been followed to implement the AIS algorithm in an organized MATLAB software package that could be easily expanded or modified in the future and is described in this document. The code utilizes proprietary software designed by Cochlear Corporation to stream pulse sequences to their brand of CIs. Once the strategy was implemented and verified with a CI emulator, an experiment to evaluate the effectiveness of AIS was performed with the participation of two adult postlingually deafened CI users. The results of these trials suggest potential directions for future research in CI sound processing.

1.3 Thesis Organization

This thesis describes research conducted during the 2011-2012 school year on cochlear implant (CI) sound processing. The work is organized in this document as follows:

- Chapter 2 provides an overview of basic audiology and introduces the fundamentals of CI operation.

- Chapter 3 describes the details of generating an electrode stimulation pattern based on sound input. Continuous Interleaved Sampling (CIS), the most common approach used in CI devices today, is discussed first. Asynchronous Interleaved Sampling (AIS) is then introduced as a potential alternative.

- Chapter 4 outlines the development and verification of the physical implementation of AIS used in the experiment.

- Chapter 5 describes the Schroeder Phase signals and how they are used in experimental trials to test CI users' ability to perform phase discrimination.

- Chapter 6 summarizes and discusses the results of the experimental trials.

- Chapter 7 summarizes the thesis and concludes the report as well as presenting several options for extending this research.

CHAPTER 2

BACKGROUND

2.1 Natural Hearing

The human auditory system collects acoustic stimulation from the environment and converts it to an electrical signal that is interpreted by the central nervous system (CNS) and eventually perceived as sound. Sounds anywhere from 20 Hz to 20 kHz can be detected by the average listener. The outer ear collects acoustic energy and transfers it to the cochlea, which performs spectral analysis on the mechanical waves. Hair cells located inside the cochlea convert the mechanical waves to electrical neural impulses, which are collected by the auditory nerve and transported to the CNS for processing. Throughout this process, both spectral and temporal information about the stimulus are conveyed to the CNS [6].

This system is analogous to a multichannel digital radio receiver consisting of an antenna, filterbank, and analog-to-digital converter (ADC). The antenna collects radio frequency (RF) energy like the ear, the filterbank performs spectral analysis like the cochlea, and an ADC converts the resulting analog signal into digital bits like the hair cells convert mechanical waves into neural impulses. Figure 2.1 illustrates the analogy between the flow of natural hearing and a radio receiver.

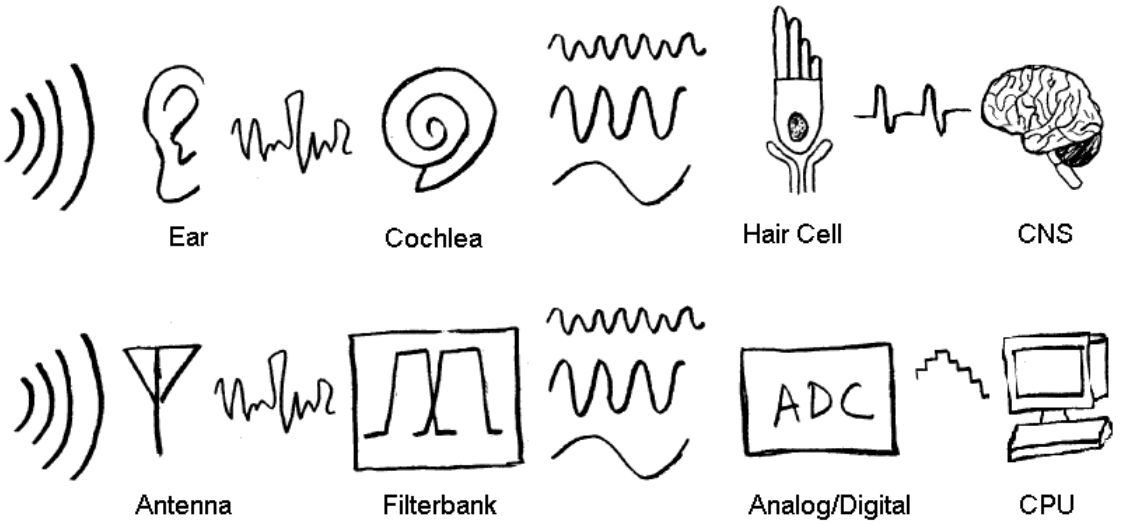

Figure 2.1: Natural hearing flow: a radio receiver analogy

The rest of this section briefly describes the details of human audition with the radio receiver analogy in mind.

2.1.1 Outer Ear: The Antenna

The outer ear is the interface between the delicate inner ear and the outside environment. Its primary purpose is to collect and conduct acoustic energy towards the cochlea, much like a radio antenna collects RF energy. The acoustic impedance of air is much lower than that of the viscous fluid inside the cochlea, so the outer ear is shaped to match the two to reduce the amount of energy that is reflected away. In the same way that a musical horn flares out to better project its sound, the outer ear flares in to better collect sound.

Once the sound waves have been conducted into the auditory channel and impinge upon the tympanic membrane, the three small bones of the inner ear complete the

impedance match. The malleus, incus, and stapes form a lever system that transfers pressure distributed across the wide tympanic membrane to the small opening of the cochlea. Figure 2.2 shows these main components of the inner ear, with the semi-circular canals normally above the cochlea omitted for simplicity. This transmission incurs very low loss, much like an electrical antenna's impedance matching network.

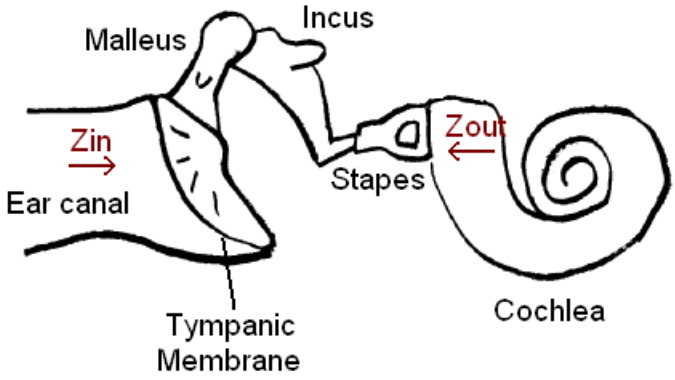

Figure 2.2: Mechanical impedance matching network in the ear

2.1.2 Cochlea: The Filterbank

The cochlea is a small spiral-shaped bone inside the inner ear that physically separates incoming sound waves based on their frequencies. If the cochlea were unraveled, it would look like a tube roughly 34 mm in length. The cochlea is filled with a thick fluid and is separated into two chambers by the basilar membrane. Incident sound waves collected by the middle and outer ear cause the basilar membrane to vibrate, conducting a travelling wave down the length of the cochlea.

Different portions of the basilar membrane resonate at different sinusoidal frequencies due to variations in the membrane's properties along its length. The basilar

membrane is stiff near the cochlea's entrance and becomes wider and more flaccid deeper inside the cochlea. As a result, high frequency waves die out closer to the cochlea's entrance and lower frequency waves travel further down the basilar membrane. A traveling wave resonates on the membrane just before dying out, causing the auditory nerve to be stimulated in a very specific location for its frequency. The CNS can then perceive different tones based on where the nerve was physically stimulated.

In essence, the cochlea is a filterbank that separates incoming sound signals spatially due to their frequency content. Figure 2.3 illustrates this concept. The frequency resolution of the cochlea is very fine, limited only by the number of individual sites where the auditory nerve can be stimulated by resonant traveling waves. Each stimulation site roughly corresponds to a group of hair cells.

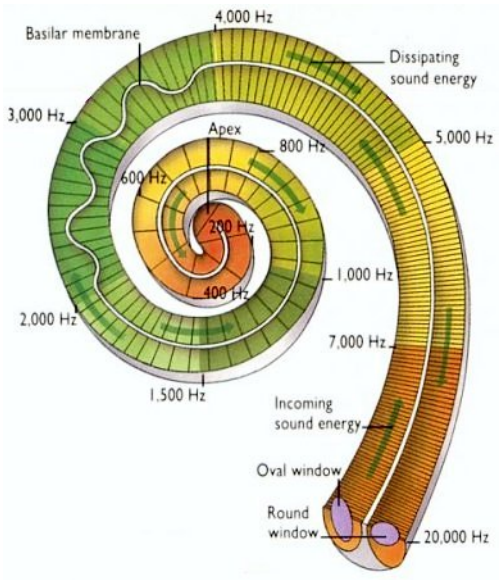

Figure 2.3: The cochlea's frequency selectivity [7]

2.1.3 Hair Cells: The Converter

Once the cochlea has separated the frequency components along its basilar membrane, hair cells complete the transfer of information to the CNS. When the basilar membrane is excited by a traveling wave, hair cells attached to the membrane are subjected to a shearing force. This shearing force opens mechanically-gated ion channels in the hair cells, which depolarizes nearby neural cells and induces a neural oscillation. There are over 10,000 hair cells in the human cochlea arranged in 3 or 4 rows, resulting in a high spectral resolution [8].

This neural impulse is known as an action potential (AP), and propagates down the auditory nerve into the CNS for higher level auditory processing. The AP is an all-or-none response, and always has the same canonical form like a digital bit in a computer system. Neurons synchronize the firing of these pulses with the peak amplitudes of acoustic stimulation, leading to the theory that the timing of APs encodes temporal or phase information [6].

The hair cells of the profoundly deaf are either severely damaged or completely missing, so the conduction of mechanical waves into the inner does not result in a neural response. The primary goal of the cochlear implant is to present this missing electrical stimulus to the CNS to provoke the perception of sound.

2.2 Cochlear Implants

The cochlear implant bypasses the inner ear through direct electrical stimulation of the auditory nerve based on sound input. A full CI system consists of an external unit and an internal unit. The external unit contains a microphone, a sound processor, and a wireless transmitter. The internal unit consists of a wireless receiver and a long,

flexible electrode array that is inserted into the cochlea. A full CI system is illustrated in Figure 2.4.

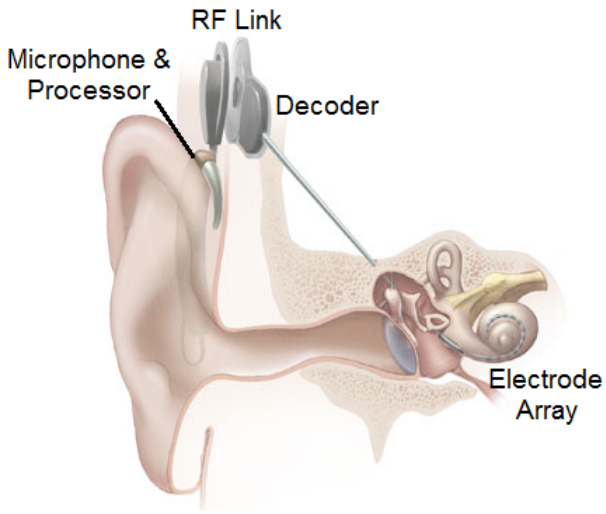

Figure 2.4: Cochlear implant system [9]

The external or behind-the-ear (BTE) unit is worn is responsible for sound processing and communication with the implant. The BTE is typically battery powered and wirelessly delivers power and instructions to the internal unit through an RF coil link. External sounds are recorded by the BTE's microphone and sampled at 16 kHz so that they can be processed by a digital sound processor and pulse patterns can be sent to the implanted unit.

The internal unit receives and decodes signals from the external sound processor and delivers these signals as electrical current pulses to the electrode array housed inside the cochlea. These current pulses are biphasic, meaning both a positive and negative current are delivered in one pulse, with a small pause in between. The

duration of the break between currents is known as the phase gap, while the duration of the currents is known as the phase width.

Cochlear Corporation's Nucleus implant system uses 22 electrode stimulation sites and thus divides an incoming sound signal into 22 spectral analysis channels [10]. An electrodogram is a visualization of electrode stimulation that plots pulses against time across all the analysis channels. Each pulse is represented as a vertical line and its relative current amplitude is denoted by its height. A simulated electrodogram for the spoken word "hello" is presented in Figure 2.5.

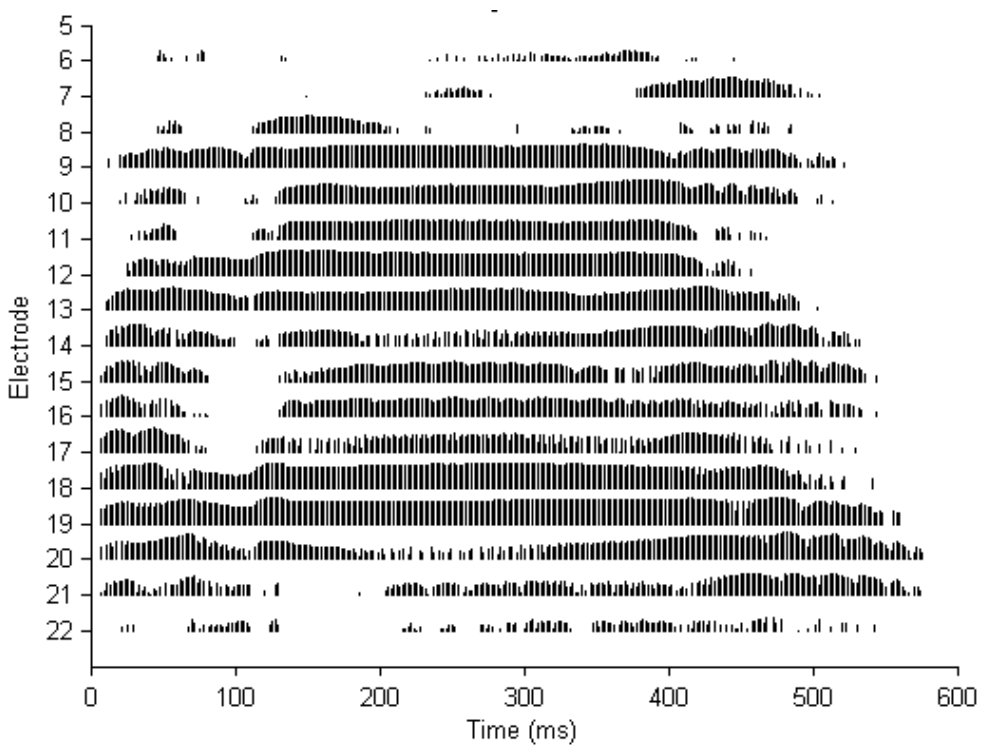

Figure 2.5: Electrodogram for the word "hello"

Filter Number	Crossover Frequency (Hz)
0	188
1	313
2	438
3	563
4	688
5	813
6	938
7	1063
8	1188
9	1313
10	1563
11	1813
12	2063
13	2313
14	2688
15	3063
16	3563
17	3063
18	4688
19	5313
20	6063
21	6938
22	7938

Table 2.1: Typical crossover frequencies for Cochlear brand implants [10]

The higher-numbered electrodes correspond to higher frequency channels while lower-numbered electrodes correspond to lower frequency channels. Table 2.1 lists the crossover frequencies of each channel used in the analysis filterbank. The crossover frequency of a channel is defined as the point where gain from the channel's filter becomes equal to the gain of its adjacent channel.

Speech processing strategies can be classified by how they convert sound input to electrode stimulation patterns. All of these strategies must address the electrode interaction problem: if two electrodes are stimulated simultaneously, their current

fields interfere and result in an undesired stimulation field. The most common way to address this issue is to interleave pulse trains between channels at a constant rate, or through Continuous Interleaved Sampling. The method explored in this thesis, Asynchronous Interleaved Sampling, uses a race-to-fire approach to ensure that no two channels fire simultaneously. These differences have significant consequences on how phase information is perceived, so these two strategies are explored in more detail in Chapter 3.

2.3 Economic & Social Impacts

Cochlear Implants are life-changing for their recipients, but they are prohibitively expensive. The average CI system costs $25,000, which limits access to these devices to people in the third world. Only 1% of the profoundly deaf has been granted an implant, so there is still a great need for access. Furthermore, the hard of hearing in developing countries face reduced social status and inability to perform work [11].

The Ohio State University otolaryngology department has been involved with promoting hearing in the third world. Dr. Edward Dodson is involved with Project Ear, a non-profit group in the Dominican Republic dedicated to donating CIs to locals to as well as training local doctors to perform implantation surgery. Dr. Dodson travels to the Dominican twice a year to perform implants for charity, typically implanting about five patients per trip. Local clinics can then take care of the patient rehabilitation process. The program has been a success, but with one snag: access to implants has been very difficult, and the program relies exclusively on cochlear implant company donations. The economics of CI distribution have been studied by

OSU business students, who are engaged in projects for social entrepreneurship in this area [11].

This thesis' research explores an important deficiency of current CI technology: the poor quality of the electrical stimulus. The CI only has 22 electrodes as opposed to the 3,000 or so stimulation sites of the natural cochlea. Postlingually deaf patients can often learn to successfully reinterpret this stimulus, but those who are born deaf have less opportunity to develop their auditory systems. An improvement in the stimulus signal, such as finer temporal resolution, would help the language development of these prelingually deaf patients.

CHAPTER 3

PROCESSING STRATEGIES

The goal of a cochlear implant speech processing strategy is to convert incoming sound into electrical pulses that can be delivered by an electrode array in the cochlea. Both stimuli generation methods described in this chapter avoid electrode interaction by ensuring that no two electrodes are activated simultaneously, but their approaches differ greatly in timing. Continuous Interleaved Sampling (CIS) stimulates each channel at a constant rate so that pulse trains from all channels can be easily interleaved together. Asynchronous Interleaved Sampling (AIS) uses a race-to-fire strategy that ensures only the most important channel is stimulated in an analysis block. The time to firing can vary between pulses in AIS, allowing each channel's stimulation rate to adapt to the timing of the signal, just as neurons can change their firing rate.

3.1 Continuous Interleaved Sampling

CIS and its variations are the most widely used CI sound processing strategies today. Analysis of a real-time audio signal is divided into continuous input blocks at a frequency known as the analysis rate. The audio signal is sampled with an analog-to-digital converter and a block of this audio input is passed through the processor's digital signal processing chain. The block diagram in Figure 3.1 shows the signal flow.

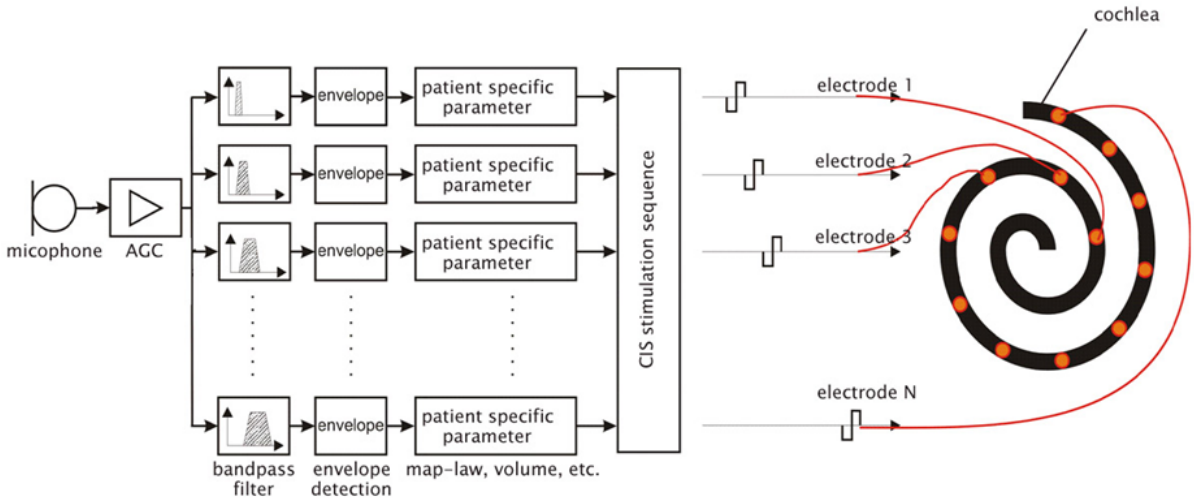

Figure 3.1: CIS block diagram [12]

The steps in CIS audio processing are [10]:

1. Use an FFT filterbank to divide audio into its channels
2. Extract each channel's power envelope
3. Map the envelope to a current level
4. Stimulate the channels in electrode order

For the implants studied in this thesis, the audio sampling rate is 16 kHz and the analysis block length is 128 samples, resulting in an analysis window width of 8 ms. The overall implant stimulation rate, however, is limited to 14.4 kHz due to safety considerations. The sampled audio is windowed with a 128-point Hann function, defined as:

$$w(t) = 0.5\left(1.0 - \cos\left(\frac{2m\pi}{L}\right)\right) \quad m = 0, \ldots, L-1 \tag{3.1}$$

This windowing operation reduces edge effects when the signal is analyzed by an FFT filterbank. After the signal is split into its component spectral channels, each

channel's average power is extracted. One version of CIS known as the Advanced Combinational Encoder (ACE) elects to stimulate only the M highest-energy channels for each analysis block to reduce overall implant energy consumption. The envelope levels of the M selected channels are converted to electrode current levels based on the patient's MAP [10].

One electrical pulse is delivered to each active channel every analysis period, resulting in an equal stimulation rate for every channel. The channels fire in succession, giving rise to an interleaved pattern like the one in Figure 3.2. Because the pulses interleave, current is never delivered to two electrodes at the same time and electrode interaction is avoided.

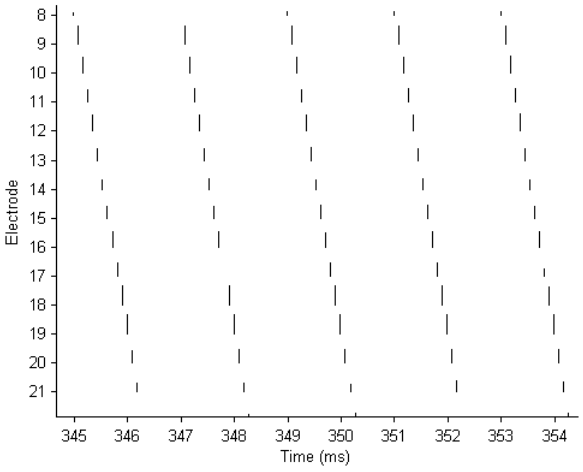

Figure 3.2: Synchronously interleaved current pulses

Neuronal activity tends to synchronize with electrical stimulation, so the main temporal signal CIS presents to the nervous system is the constant implant stimulation rate. This carrier frequency is completely uncorrelated with the signal's temporal characteristics, usually resulting in poor phase perception [1].

3.2 Asynchronous Interleaved Sampling

AIS has been proposed as an alternative way of stimulating the cochlea by Sit et al [1]. Unlike in CIS, the stimulation rate is not synchronous so the spacing between pulses is allowed to vary. This is accomplished by associating one integrate-and-fire neuron with each channel which charges at a rate proportional to the strength of the audio signal in that channel. These neurons then begin a "race-to-spike." The first neuron to charge to a threshold value "fires" (stimulates its electrode) and the race starts again. The winning channel is inhibited for some time to prevent it from completely dominating the other channels, imitating the absolute refractory period of neurons. A block diagram of AIS is shown in Figure 3.3.

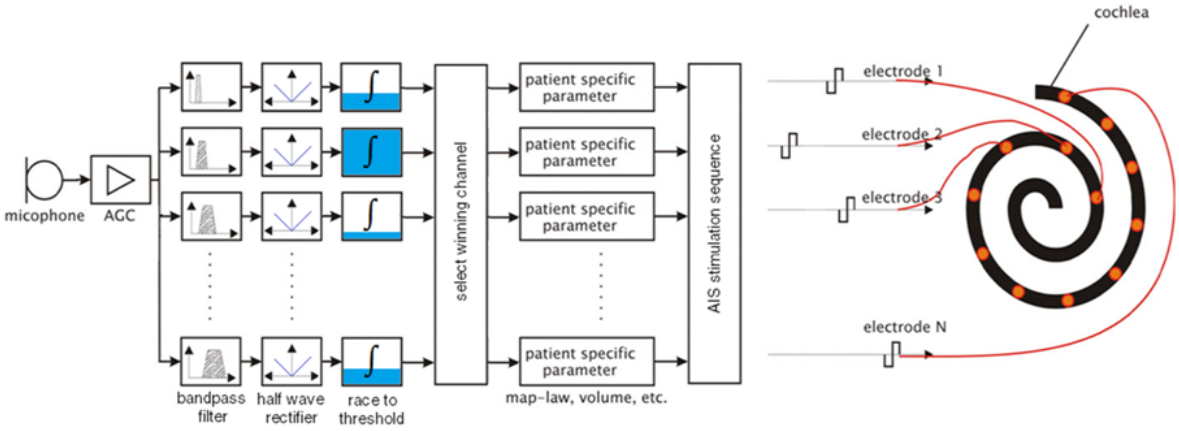

Figure 3.3: AIS block diagram [12]

To summarize, the steps in AIS processing are:

1. Use a filterbank to divide the audio signal into different channels. This front-end could be implemented as an analog or digital system.
2. Half-wave rectify each channel.
3. Begin integrating the rectified channels to charge their associated neurons, initiating the "race-to-spike."
4. The neuron that reaches a set threshold first "wins" and stimulates its channel with a current pulse proportional to the channel's envelope.
5. The winning channel is inhibited with by some function which gradually rolls off with time to prevent its neuron from winning again immediately.
6. All the neurons are reset and the race starts again from Step 3.

The integration operation of each neuron effectively low-pass filters the rectified channels, implementing a sort of envelope detection common to all CI processing strategies. The maximum channel stimulation rate is therefore dictated by the inhibition function used to penalize them. The inhibition function in this thesis features an exponential rolloff modeled by the Fermi-Dirac equation [1]:

$$I_{inh}(t) = \frac{A_{inh}}{1 + e^{k(t-\tau_{inh})}} \tag{3.2}$$

The inhibition level is I_{inh} (which can be abstracted as an electrical current preventing a neuron from charging) and A_{inh} is the maximum inhibition level in simulation units. The function can be tuned by two parameters: k controls the steepness of the function's rolloff and τ_{inh} is the time when inhibition becomes one-half of its original value. Increasing k quickens rolloff while increasing τ_{inh} decreases the maximum channel stimulation rate by lengthening each channel's refractory period.

The inhibition current function is plotted in Figure 3.4 for $\tau_{inh} = 1$ ms to mimic the relative refractory period of a neuron and $k = 5 \times 10^3$, 4×10^4, and 5×10^5. In this project's implementation of AIS, k was selected to be 10^4 since that steepness enforces some channel inhibition while still allowing a very strong channel to fire again if necessary.

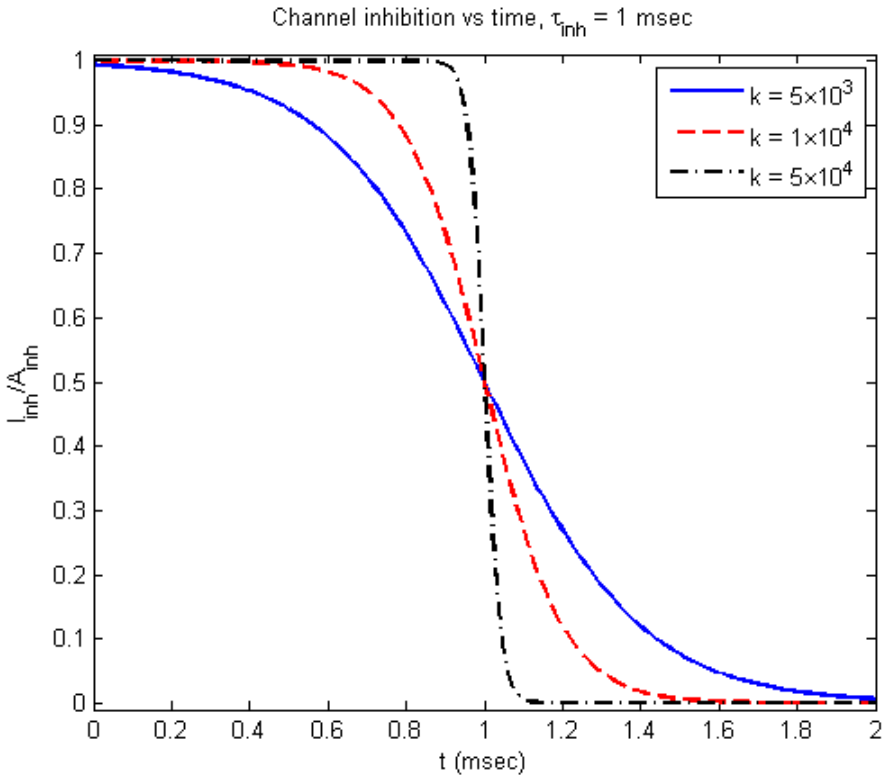

Figure 3.4: Inhibition function with exponential rolloff

The algorithm is laid out in Figure 3.2 and is directly taken from Sit et al. [1]. Since AIS was implemented in a sampled system for this thesis, all values of t described

in the algorithm are discrete timesteps. The actual code used in implementation is outlined in further detail in Chapter 4.

Initialize channel voltages $V_{ch}(t) = 0 \,\forall$ channels
Initialize time-of-last-spike $t_{lastspike} = -\infty \,\forall$ channels
Initialize spiking output $spike(t) = 0 \,\forall$ channels
for each timestep t **do**
  for each channel **do**
    Set $penalty(t) = I_{inh}(t - t_{lastspike})$
    Increment V_{ch}+= $max[HWR_{ch}(t) - penalty(t)]$ (Step 3)
    $\{HWR_{ch}(t) =$ half-wave-rectified output for that channel (Steps 1 & 2)$\}$
  end for
  if $\max_{\forall\, channels}[V_{ch}(t)] > V_{thresh}$ **then**
    Find $\max_{ch}$ = channel whose $V_{cap}(t) = \max_{\forall\, channels}[V_{ch}(t)]$
    Set $spike(t) = E(t)$ in channel max_{ch} (Step 4)
    Set $t_{lastspike} = t$ in channel max_{ch} (Step 5)
    Reset $V_{ch}(t) = 0 \forall$ channels (Step 6)
  end if
end for

Figure 3.5: AIS pseudocode

The channels fire pseudorandomly, giving rise to an interleaved pattern like the one in Figure 3.6. Because only one channel is allowed to fire at a time, electrode interaction is avoided. Since the time between pulses varies, the nervous system does not synchronize with the implant stimulation frequency as it does in CIS. This results in a stimulus that is better correlated with the signal's temporal and phase characteristics [1]. To determine whether this would translate into CI user phase perception, AIS was tested with CI users in an experiment described in Chapter 5.

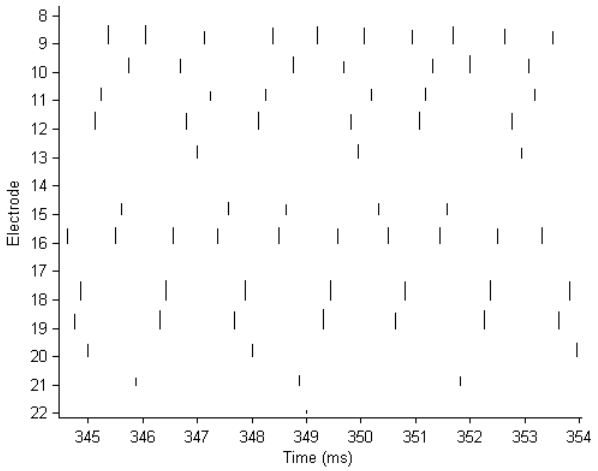

Figure 3.6: Asynchronously interleaved current pulses

CHAPTER 4

SYSTEM DESIGN & IMPLEMENTATION

Since AIS is a novel stimulation technique, it is not in use in any commercial implants and had to be physically implemented before it could be tested. A proprietary software library for CIS, called the Nucleus MATLAB Toolbox (NMT), was available from Cochlear Corporation. The NMT was modified to generate AIS pulse sequences which could then be transmitted to an implant via the hardware setup. Once AIS was implemented, the output was verified with an oscilloscope before testing the system with implant users.

4.1 Hardware

The hardware used to stream pulse sequences to the patient is produced entirely by Cochlear Corporation, and the equipment's setup is shown in Figure 4.1. A laptop computer running MATLAB is used to generate pulse sequences and stream them to the programming pod, which connects a Nucleus speech processor to the computer through a USB port. The speech processor encodes the pulse sequences into an RF communication signal which are delivered by the transmitter through a wireless inductive link to a CI. The transmitter is affixed to a volunteer's CI or a

physical emulator with a small magnet that aligns the transmitting and receiving RF coils.

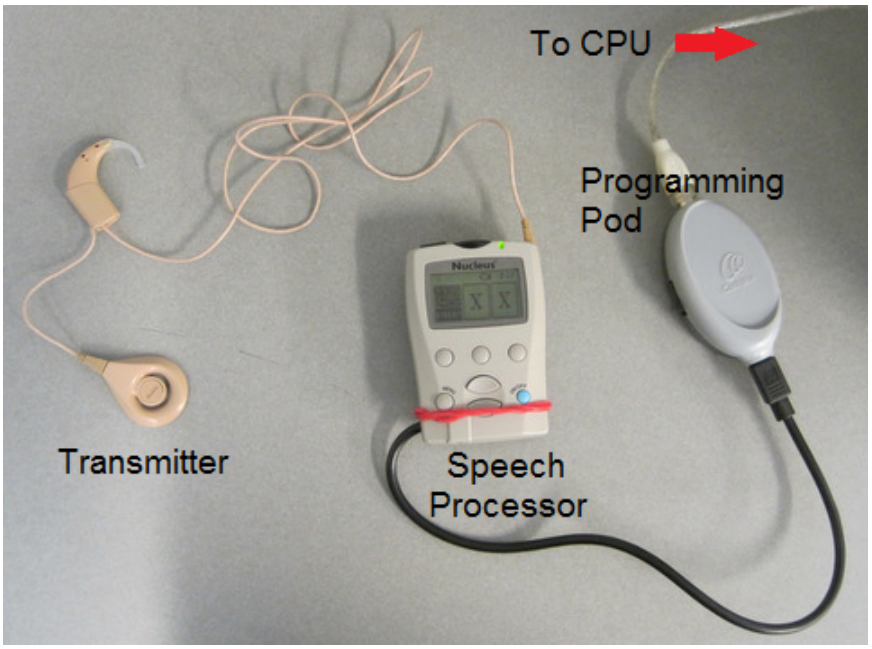

Figure 4.1: CI streaming hardware setup

This hardware setup was designed to work with the NMT, but other software configurations were available. NMT was chosen for this project due to the author's familiarity with MATLAB.

4.2 Software

All software used in this project was written in MATLAB. The Nuclear MATLAB Toolbox (NMT) is a proprietary software library that allows scientists to perform sound processing in MATLAB and transmit the results through this test equipment.

Since the NMT is carefully engineered and easy to maintain, all AIS functions were written in the style of the NMT to keep the software organized.

4.2.1 Nuclear MATLAB Toolbox

The NMT is a proprietary software library of MATLAB functions that implement algorithms used in Nucleus CI systems. It was originally designed to achieve a number of goals [13]:

- **Specification** for signal processing algorithms. Writing a program to specify an algorithm leads to less ambiguity than pseudocode or pure mathematics.
- **Education** in the area of signal processing algorithms, taking advantage of MATLAB's powerful visualization capabilities.
- **Development** of new signal processing algorithms. MATLAB is a higher level language and better suited for iterative testing than a digital signal processor's assembly language.
- **Experimentation** with new speech processing strategies. Streaming functions are used to send MATLAB-generated stimuli to an implant. This allows audio to be processed by new algorithms and enables perceptual experiments to be developed in MATLAB.

In this thesis, the NMT was used to develop the first implementation of AIS in a commercially available implant system (Cochlear International's Nucleus and Freedom systems). NMT and MATLAB code was also used to administrate the psychoacoustic tests described in Chapter 5. The code in this document could also be

considered a concrete specification for the AIS algorithm as well as a useful pedagogical tool for describing AIS to researchers unfamiliar with it.

The heart of the NMT is the `Process` function. `Process` is set up so that individual processing functions can be chained together and called as a unit. All of these processing functions are suffixed with "`_proc`" and have the structure shown in Listing 4.1:

```
function v = Example_proc(p, u)     % function definition
switch nargin                     % switch on number of input arguments
case 0:                           % Set default parameters
case 1:                           % Calculate parameters
case 2:                           % Perform processing
end;
```

Listing 4.1: Generic processing function structure

A piece of code that would process a sampled audio signal from the file '`hello.wav`' with a CIS map is shown in Listing 4.2. This function will run through the process list of the struct `p`, which is the default CIS map, and return the result `q`, which is a pulse sequence struct.

```
p = CIS_map;
q = Process('hello.wav', p);
```

Listing 4.2: Processing a .wav file in NMT

A sequence struct contains all the information the speech processor needs to generate a wireless transmission of a series of current pulses to the implant. The code in Listing 4.3 is an example of streaming a sequence to an implant through the hardware system. A `client` object is created that represents the streaming device in the code, and it is invoked to stream the pulse sequence `q`.

```
client = NICStreamClient;                % create client
client = initialiseClient(client,'L34-CIC4-1');

client = sendData(client, q);            % set sequence to be transmitted
client = startStream(client);            % start streaming
pause(2);                                % wait until streaming finishes
client = stopStream(client);             % stop streaming
```

Listing 4.3: Streaming a sequence to an implant

Using the NMT, pulse sequences that are generated using novel sound processing strategies can be streamed to a user. This algorithm would be organized as a list of processes that can be packaged inside a MAP structure along with critical parameters.

4.2.2 AIS MATLAB Implementation

The main product of the AIS MATLAB implementation is the structure AIS_map. This structure contains all parameters and processes necessary to perform AIS processing on a sampled waveform. The processes used in AIS are listed in Table 4.1 in order of execution along with brief descriptions. Processes with an underscore suffix are proprietary NMT functions that were modified to suit the purposes of the AIS implementation.

Process Name	**Description**
`Wav_proc`	Converts `'.wav'` file to a vector
`FIR_filterbank_proc`	Breaks signal into multiple analysis channels using an FIR filterbank
`HWR_proc`	Half wave rectifies each analysis channel
`Integrate_fire_proc`	Generates pulse magnitude trains through the race-to-fire method
`LGF_proc_`	Scales pulse magnitudes into relative current levels using a logarithmic loudness growth function
`Channel_mapping_proc_`	Maps relative current levels onto a physical stimulation sequence

Table 4.1: Process list for AIS analysis

`Integrate_fire_proc` is the only non-NMT function used in the AIS implementation. This function performs the algorithm outlined in the pseudocode listed in Figure 3.2. The essential parameters of this process match the AIS parameters outlined in Chapter 3 and are tabulated in Table 4.2. The timing parameters each have an associated `_avg` and `_dev` value, corresponding to a mean and standard deviation value for the parameter if randomization is desired.

Parameter	**Description**
`A_inh`	Maximum amplitude of inhibition level
`k`	Rolloff steepness parameter
`tau_avg`	Average value for inhibition current half-life τ_{inh}
`tau_dev`	Standard deviation for inhibition current half-life τ_{inh}
`threshold_avg`	Average value for spiking threshold V_{thresh}
`threshold_dev`	Standard deviation for spiking threshold V_{thresh}

Table 4.2: Parameter list for `Integrate_fire_proc`

The MATLAB code implementing `Integrate_fire_proc` is in Appendix A. Further code used to administrate the psychoacoustic experiment described in Chapter 5 is also in Appendix A and a portion of the code uses the auditory AUX scripting language designed by Kwon [14].

4.3 Verification

Once the system was generating an RF output, the output was tested with an implant emulator. The emulator consists of a receiving coil, decoder circuitry, and a female DB-25 connector output terminal. Because the emulator output is a current, an oscilloscope could not directly measure the output of the emulator. Therefore, an

interface circuit that could convert the emulator's current output to a voltage was designed. The interface circuit schematic is shown in Figure 4.2.

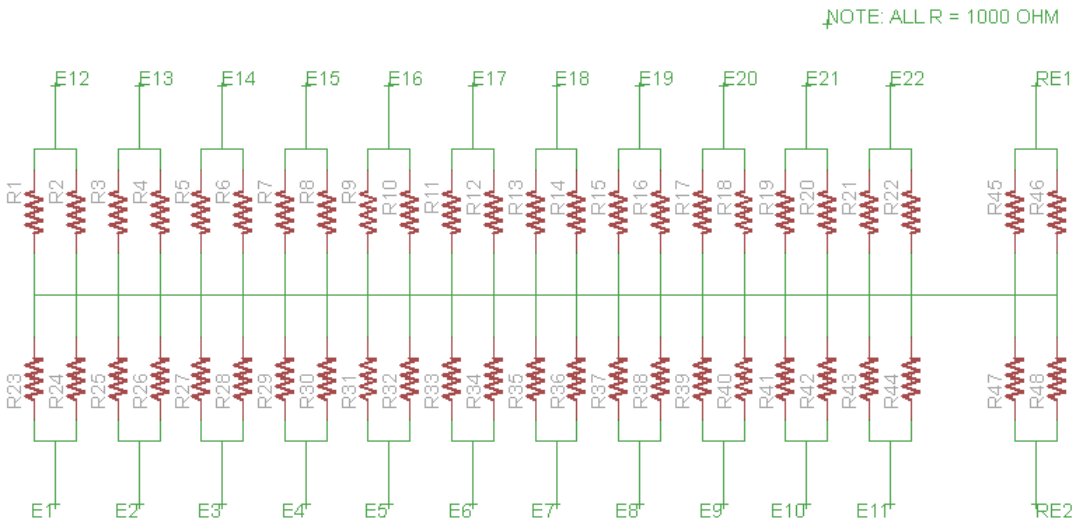

Figure 4.2: Emulator oscilloscope interface circuit

Two electrodes at the output of the emulator are active at any one time: one channel electrode (labeled E1-E22) and one reference electrode (labeled RE1 and RE2). The emulator current output (rated maximum of 1.671 mA) passes through a 1 kΩ resistance network, converting the current into a voltage value that can be measured by the oscilloscope probes. Every resistor in the schematic is valued at 1 kΩ, so any pathway between two electrodes has a resistance equal to 1 kΩ.

The circuit was assembled and soldered on a piece of perforated circuit board. A male DB-25 connector's terminals were wired to the circuit to interface with the emulator. The DB-25 ports 1-22 were connected to the circuit nodes labeled E1-E22 and ports 24 and 25 were connected to RE1 and RE2 respectively. The finished interface board is shown connected to the implant emulator in Figure 4.3.

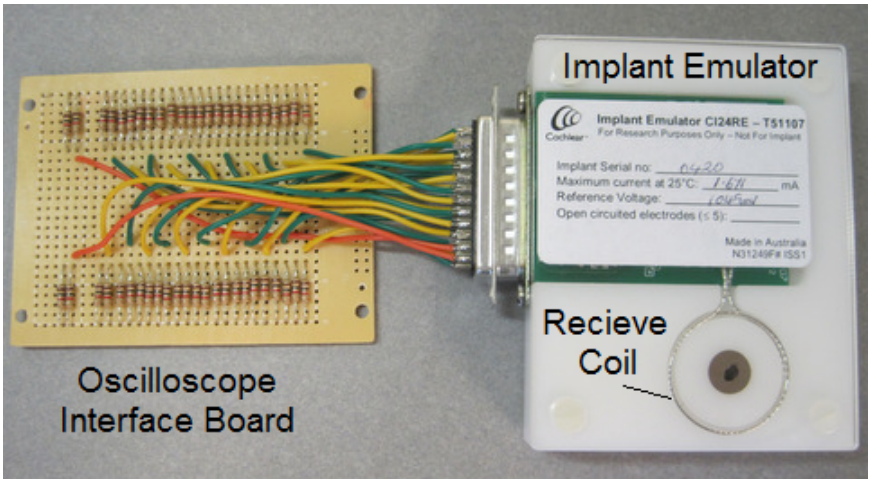

Figure 4.3: CI emulator and interface board

The setup pictured in Figure 4.1 was used to stream CIS and AIS pulse sequences to the emulator. An oscilloscope was attached between two electrodes on the interface board to measure the decoded current strengths. Electrical pulses appeared with the expected phase widths and phase gaps as shown in Figure 4.4. With verification of the system implementation complete, the perceptual experiment could be designed and carried out.

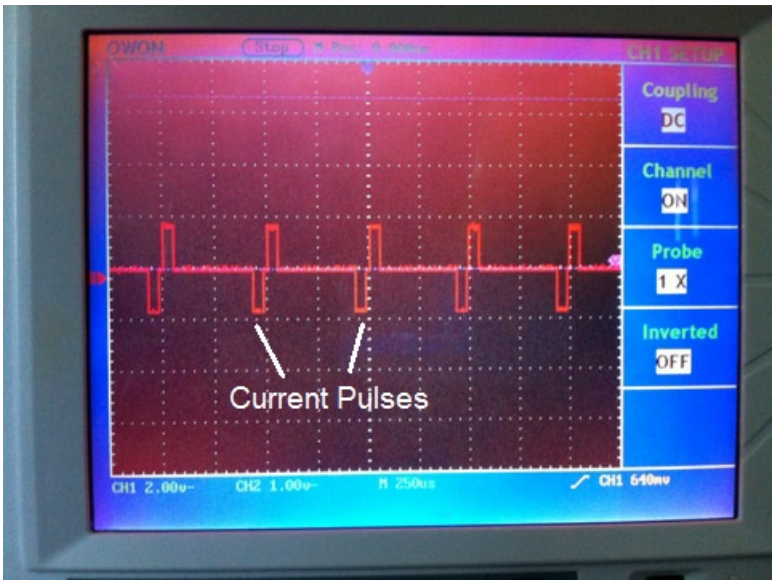

Figure 4.4: Oscilloscope measurement of current pulses

CHAPTER 5

EXPERIMENTAL DESIGN

Once AIS was physically implemented, psychoacoustic tests were performed on a couple of CI users to measure the algorithm's success in presenting phase information. The experimental sound stimuli were tuned such that their only differences were temporal in nature, avoiding any confounding spectral cues. AIS was to be compared with a version of CIS called Advanced Combinational Encoding (ACE), where a subset of 12 channels are stimulated during each analysis block.

5.1 Stimuli: The Schroeder Phase Signals

Georg Ohm once famously postulated that the human ear is incapable of understanding the relative phase of different signals [15], and this belief prevailed in the audiology community for over a century. However, a result published in 1970 by Dr. Manfred Schroeder, who was working at Bell Labs on radar systems at the time, renewed interest in the importance of phase in hearing [16]. A major problem in radar systems were that multiple reflections could constructively interfere at the radar's receiving antenna and result in high voltage peaks with the potential to damage sensitive receiver circuitry. In order to better understand this problem, Schroeder

explored ways to construct signals with the lowest peak amplitudes possible. The resulting waveforms are known today as the Schroeder phase signals.

The Schroeder phase signals are periodic tones that consist of the sum of a fundamental frequency and its first N harmonics. These harmonics are added with equal amplitude and with Schroeder Phase. In mathematical terms,

$$\text{Schr}\pm\,(t) = \frac{1}{N}\sum_{n=1}^{N}\cos(nf_0t \pm \theta_n) \tag{5.1}$$

where f_0 is the fundamental frequency and θ_n is the n^{th}-harmonic Schroeder phase:

$$\theta_n = \pm\frac{\pi n(n+1)}{N} \tag{5.2}$$

The positive Schroeder Phase signal (Schr+) uses a positive θ_n for each harmonic and the negative signal (Schr−) uses a negative θ_n for each harmonic. In this experiment, 50 harmonics were used for each phase signal (N=50). The two phase signals are plotted in Figure 5.1 with a fundamental frequency of 50 Hz.

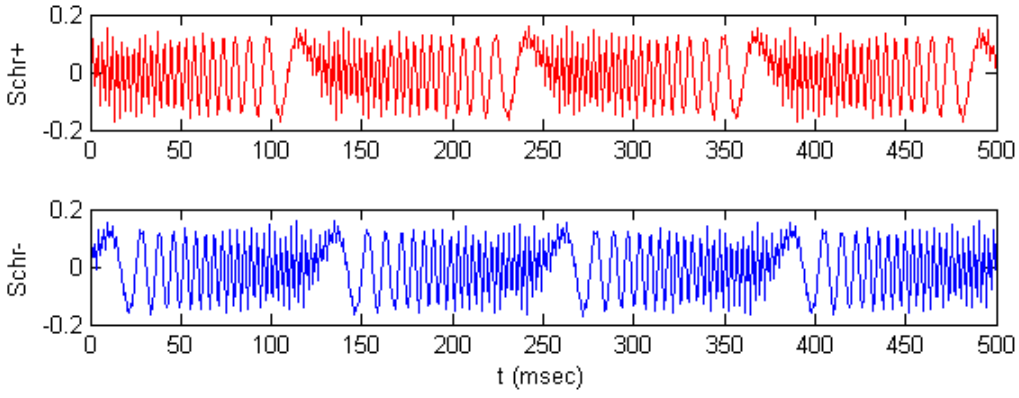

Figure 5.1: 50 Hz Schroeder phase signals

Since all harmonics are added with the same amplitude, both Schr+ and Schr− signals feature identical frequency content and only differ in their phase content. In essence, these signals are time-reversed copies of each other. This time reversal must be well-preserved by a processing strategy in order for a CI user to be able to detect a difference between the two signals. Figure 5.2 compares the electrical stimuli generated by AIS. There appears to be a strong difference in the temporal pattern between the two signals; however, experimental trials were required to determine whether or not CI users could discriminate between these signals.

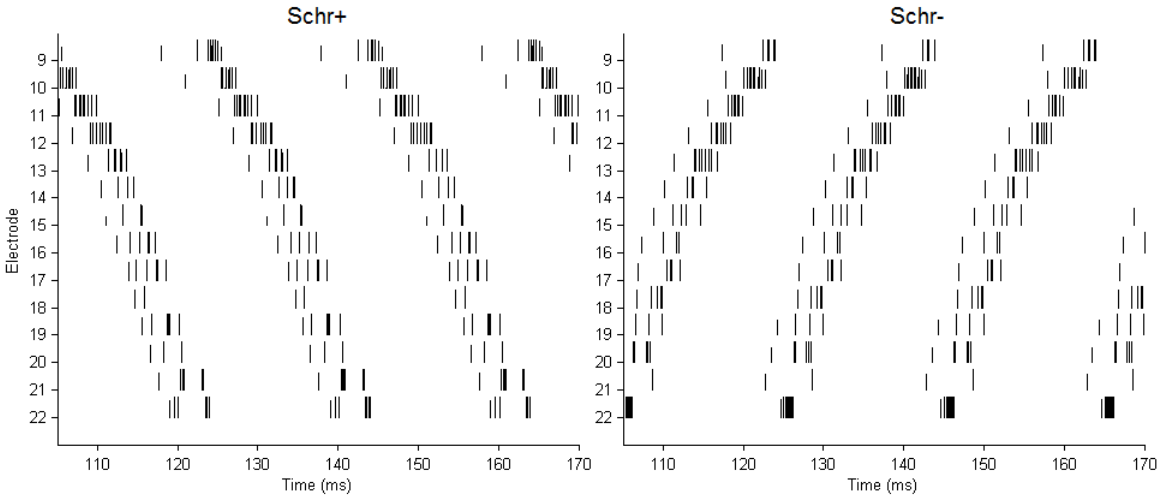

Figure 5.2: AIS processing of 50 Hz Schr± complexes

Another purpose of this experiment was to contrast AIS's ability to present these signals to CI users with CIS (or ACE). One way to predict differences between the strategies in conveying these signals is to compare their respective electrodograms, which are shown in Figure 5.3 for the 50 Hz Schr− complex. AIS tracks the peaks of the harmonics more precisely than ACE, suggesting that the AIS-generated electrical

stimulus preserves temporal information, while there is significant temporal smearing in the ACE electrodogram due to its use of synchronous stimulation.

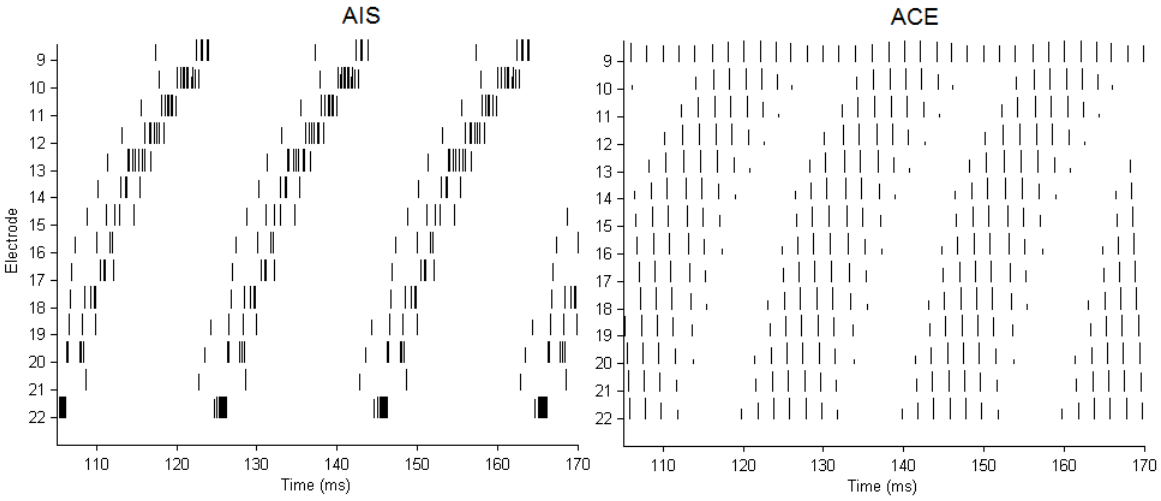

Figure 5.3: AIS vs ACE processing of 50 Hz Schr− complex

5.2 Phase Discrimination Test

A test was designed to see if CI users could differentiate between the Schr+ and Schr− signals with the CIS and AIS processing strategies. The system setup described in Chapter 4 was used to present stimuli to participants. Each stimulus was a series of two Schroeder phase complexes each with a duration of 500 ms separated by 500 ms of silence. The first tone presented was always Schr− and the second tone was randomized to be either a Schr+ or a Schr− tone. A 10 ms linear ramp was included at both the start and end of the signal to cut down on edge influences. Participants were asked to determine whether the two tones were the same or different.

An administration of the phase discrimination test involved two short training sessions and several longer testing sessions, each containing an equal number of phase discrimination trials for the fundamental frequencies under test. The fundamental frequencies of 15, 25, 50, 100, 200, and 400 Hz were all tested at some point. Training sessions consisted of 8 phase discrimination tasks presented in a random order with verbal feedback given to the user. Testing sessions consisted of 40 phase discrimination tasks without any feedback.

One of the training sessions and some of the testing sessions were performed using the participant's ACE map and the other training session and other testing sessions were performed using an AIS map. Two adult postlingually deafened adult CI users between the ages of 60-80 volunteered for participation. The threshold and comfort values for the AIS map were readjusted before performing any trials so that tones were at the same volume for each strategy. The AIS map parameters of V_{thresh} and τ_{inh} were varied throughout some of the experimental trials to explore their effects on phase perception as well.

CHAPTER 6

RESULTS

The Schroder phase discrimination (SPD) test was administered to two adult CI patients in several exploratory experimental trials. These preliminary results are part of the first wave of AIS experiments and suggest new directions for future studies in temporal cue recognition in CI users. The parameters of AIS were adjusted in an ad hoc fashion to explore their potential effects and are marked as AIS I-VIII. These parameter adjustments are listed in Table 6.1.

Strategy Name	$\tau_{inh}(sec)$	V_{thresh}	**Description**
AIS I	3×10^{-4}	0.35	Parameters suggested in original proposal [1]
AIS II	5×10^{-5}	0.075	Allows faster channel stimulation rates
AIS III	3×10^{-4}	2.0	More difficult to spike
AIS IV	1×10^{-2}	2.0	Slower channel stimulation rates
AIS V	3×10^{-4}	0.05	Easier to spike with lower threshold
AIS VI	1×10^{-1}	5.0	Most difficult to spike
AIS VII	3×10^{-4}	0.5	Reduced number of harmonics to $N = 25$
AIS VIII	1×10^{-1}	5.0	A 100 ms ramp was added to signal

Table 6.1: Table of tested AIS parameters

6.1 Phase Discrimination Results

The first trial's results are plotted in Figure 6.1. Fundamental frequencies of 50, 100, 200, and 400 Hz were tested. It appeared that discrimination actually improved with increasing frequency, which is counter-intuitive and conflicts with previous SPD experimental results [17]. Lower frequency complexes feature slower temporal changes, so they should be easier to differentiate than higher frequency complexes.

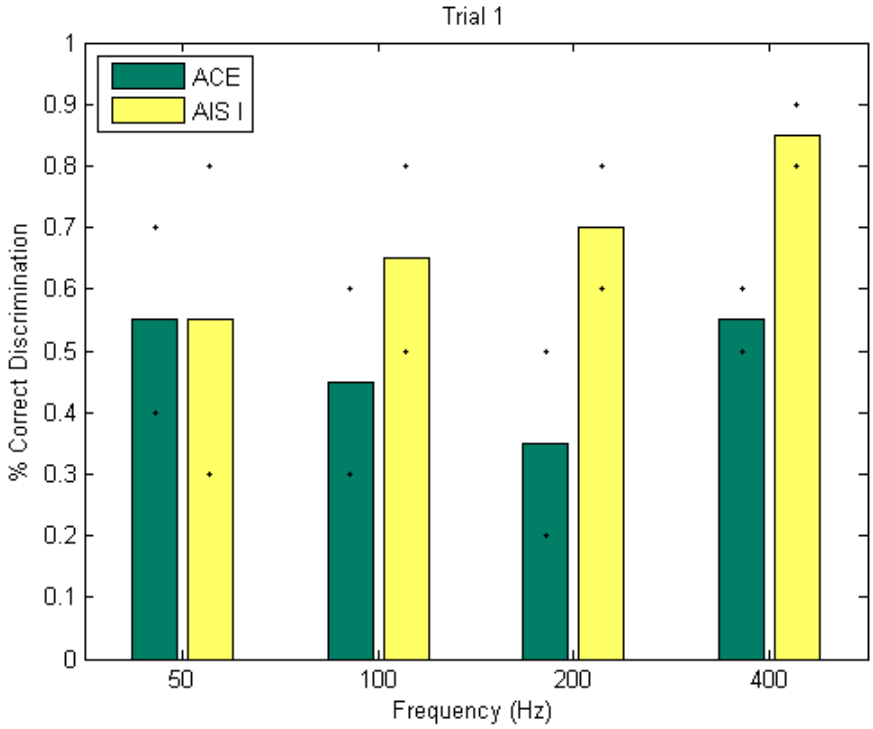

Figure 6.1: SPD results - Trial 1

Upon further inspection of the pulse sequences for the 400 Hz tone (Figure 6.2), it became clear that a spectral cue was being detected instead of a temporal cue. Since different channels were being stimulated for each signal, the subject was not

really detecting differences in phase. This strong spectral cue arose from the timing parameters (AIS I) of this trial.

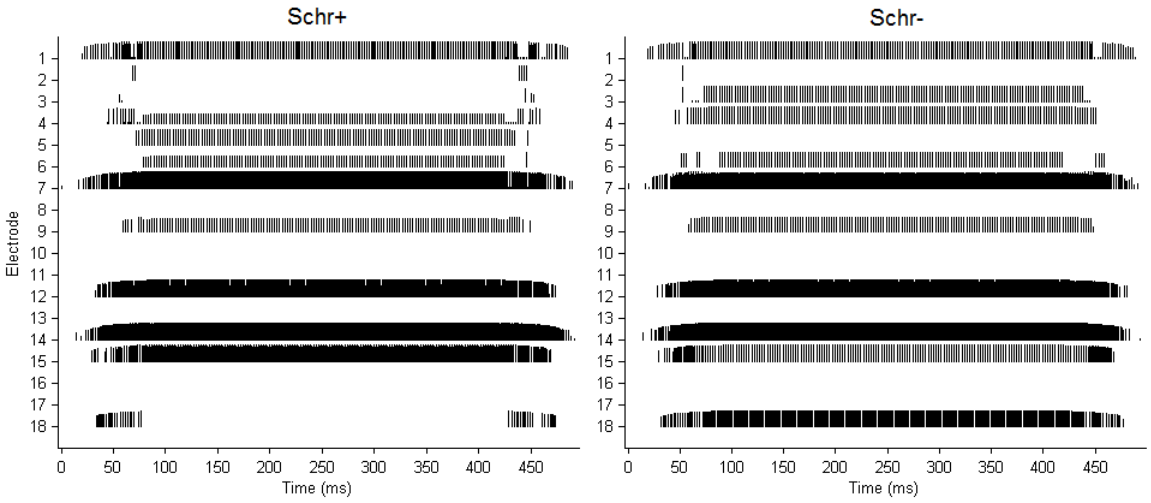

Figure 6.2: Confounding spectral cue in the 400 Hz Schroeder signals

The AIS parameters were changed to better present temporal cues. A higher spiking threshold V_{thresh} corresponds to a more sparse representation that tends to sample the analysis channels only at their peaks. Changing τ_{inh} affects the maximum channel stimulation rate in any given channel, and raising it prevents any one channel from dominating the rest.

A second trial with a second CI user was performed after some tuning of the AIS parameters (AIS II & III). Also, the fundamental frequencies of 25 and 50 Hz were tested since temporal cues are made more prominent by the slow variation of lower frequency tones. The results are shown in Figure 6.3.

No significant difference was seen between AIS and ACE for SPD, except in the 50 Hz tone. The trial participant was able to detect a difference at the onset of the

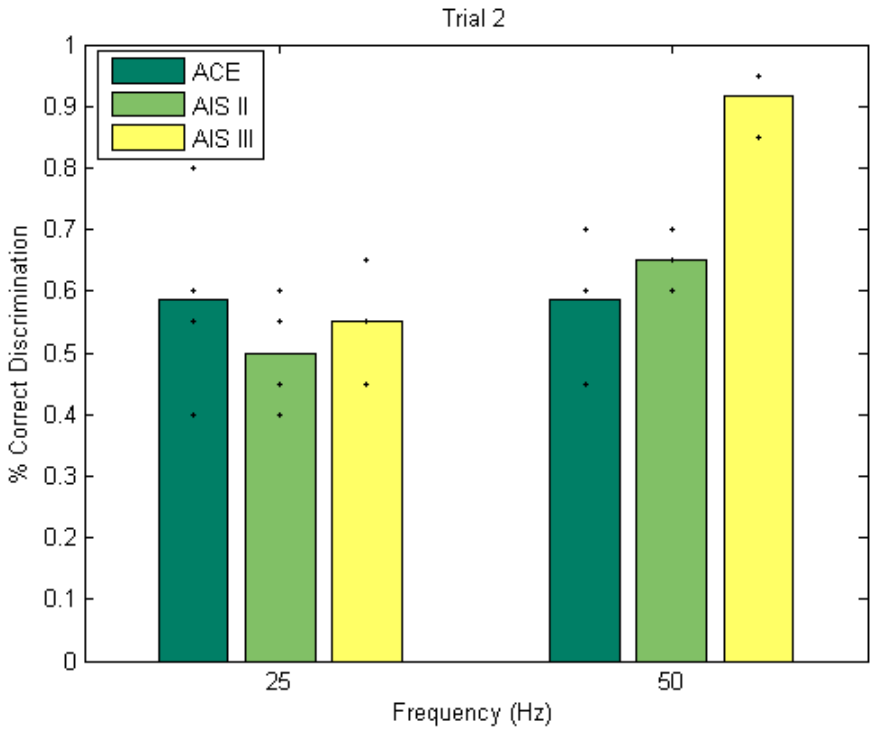

Figure 6.3: SPD results - Trial 2

Schroeder tones which arose from the 10 ms ramp window applied to the experimental stimuli. This result was unexpected, but it showed that a CI user can detect temporal cues.

Another trial was performed with this participant with AIS parameters adjusted several times (AIS IV-VIII) in an attempt to convey the main temporal variation of the Schroeder phase signals. Fundamental frequencies of 15, 25, and 50 Hz were tested. Shorter sessions with only 10 trials per frequency per parameter adjustment were used in order to facilitate the testing of more conditions.

The results showed no significant improvement of AIS over ACE. The CI user was able to detect temporal cues from the ramping of the signal edges at 50 Hz. When

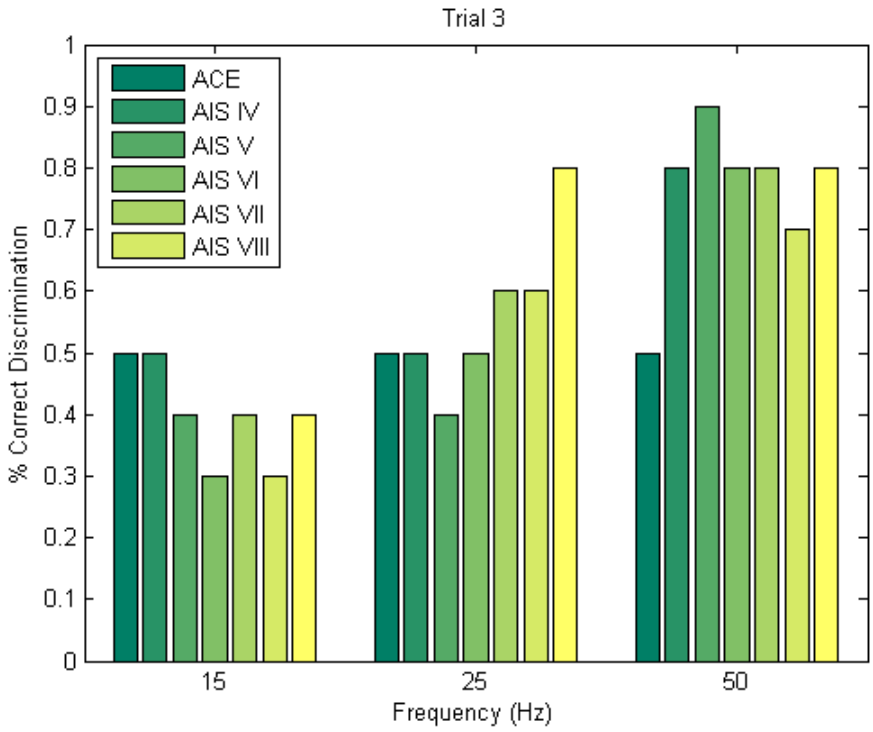

Figure 6.4: SPD results - Trial 3

the ramp's duration was extended to 100 ms, an improvement in detection resulted at 25 Hz as well. Therefore, it appeared that SPD performance did not depend on AIS parameter selection, but rather depended on the duration of the ramp function applied to the stimuli.

6.2 Discussion

The subject performed no better than chance at SPD in the first trial for lower frequencies, no matter which processing strategy was used. After a parameter adjustment, however, a second subject was able to perform SPD by detecting the temporal

ramping of the signal edges for the 50 Hz tone, averaging above 75% correct identification. It seemed that in this situation the two stimuli differed in their temporal presentation during the ramped portions, and this difference was noticeable to the CI user. When the ramp was extended, the temporal cue appeared to be present at the 25 Hz tone as well. This suggests that CI users are indeed able to detect some fine temporal cues.

However, the fine phase structure of the signal was not well-detected by the subject in Trials 2 and 3. Only the temporal cue at the signal edges was detectable. This could be because neural populations are shared between electrodes, so the Schroder signals' quick sweeping of the electrodes in time may smear across the auditory nerve, limiting temporal resolution.

These results show worse SPD performance overall than the experiments performed by Drennan et al. [18]. The CIs used in this experiment were Advanced Bionics brand implants, which feature greater electrode separation than the Cochlear brand implants used in this study. If electrode separation is increased by turning off every other channel, the temporal variations of the Schroeder phase signals might be more easily differentiated. Another difference is that the stimuli used by Drennan et al. did not use a hard number N of harmonics when constructing the tones, but rather used all the harmonics up to 5000 Hz. This resulted in a higher number of harmonics being presented to CI users for the lower frequency tones, which may have had a significant impact on SPD success.

CHAPTER 7

CONCLUSION

7.1 Summary & Conclusions

This thesis contributes the first psychoacoustic tests of Asynchronous Interleaved Sampling performed with commercially available implants. AIS was originally proposed to demonstrate the benefit of asynchronous stimulation on conveying phase information. However, the algorithm was never tested with actual CI patients, so its claim of improving phase perception has stood unsubstantiated. The software design and implementation of AIS laid the groundwork for the first qualitative tests of the method's effect on phase discrimination.

The psychoacoustic experiments performed showed no significant benefit of AIS over ACE in the perception of the Schroeder phase signals. However, factors such as electrode spacing and tone harmonic content could have played a large role in the experimental results. Even though asynchronous sampling's effect on CI processing is not yet clear, it can be investigated further in the engineering domain as a biologically inspired approach to data conversion and processing.

7.2 Future Work

The results of these initial trials motivate some future directions for research in CI sound processing and electrical engineering. Many of these ideas were raised by the thesis committee during this thesis' oral defense, and each of these projects involve a close symbiosis between engineering and biology. Five potential considerations are:

1. The AIS algorithm can be modified to use a winner-take-some approach. The problem with AIS's winner-take-all strategy is that some channels may not be stimulated at all when stronger channels dominate, ultimately discarding important spectral information. AIS could be augmented with a hysteresis threshold so that every time one channel wins, the next strongest channels above a lower second threshold fire immediately after the winner in a miniature CIS stimulation block. This small group of firing winning would then be inhibited in some way, ultimately allowing more channels to participate. This winner-take-some method is effectively a hybrid between CIS and AIS.

2. The Schroeder phase discrimination test could be repeated with every other electrode turned off. This would result in more separation between neural populations with time which may produce results more similar to other experiments [18] and provide more insight about temporal cue perception in cochlear implants.

3. There are delays inherent in the process of basilar membrane propagation that are important in understanding how fine timing information is processed in natural hearing. For example, lower frequency waves that travel further towards the apical end of the cochlea are delayed more before stimulating neurons than

higher frequency waves that resonate near the basal end. This uneven delay could also be considered in audio processing circuit design.

4. Orthogonal frequency-division multiplexing (OFDM) is used in communications systems to encode digital data streams onto multiple carrier frequencies, improving performance under severe channel conditions. Since these carriers can add constructively, high peak-to-average power ratio (PAPR) signals can result and cause difficulties in signal processing. The Schroeder phase signals exhibit the lowest PAPR possible for a harmonic complex, and may be well-suited for OFDM. A better understanding of how the auditory system processes high PAPR tones may also provide some insight in how to design better OFDM systems. Not only will a better understanding of auditory systems improve OFDM, but more likely, the rich literature on OFDM can be used to build better models of auditory systems. Understanding of auditory systems is hampered by the considerations needed to work with human subjects. Since work on OFDM communication systems requires no ethical considerations, the understanding and experimentation with OFDM technology is much deeper.

5. Other sets of orthonormal functions, besides the Fourier harmonics, may better represent the ear's natural response. Analysis with a more optimal basis set could provide more natural sound processing. The impulse response and dynamics of the basilar membrane would be important in determining this ideal basis set.

Each of these options will involve a rich dialogue between audiologists and engineers. This interdisciplinary conversation has the potential to advance the understanding of neuroscience as well as methods of signal processing and circuit design. Hope for the disabled lies entrenched in the primordial circuits of the human brain, and the integration of knowledge across modern frontiers is the key to understanding them.

APPENDIX A

MATLAB CODE

The software developed for this thesis is organized into two categories: the Asynchronous Interleaved Sampling implementation and Schroeder phase experiment administration. Portions of the Nuclear MATLAB Toolbox were used to interface with the experimental setup.

A.1 AIS_map.m

```
function p = AIS_map(p)

% AIS_map: Calculate AIS map parameters.
% To perform processing, use:
%   q = Process(p, audio)
%
% p_out = AIS_map(p_in)
%
% p_in:  A struct containing the clinical parameters.
%          Any fields omitted will be set to default values.
% p_out: A struct containing the clinical and derived parameters.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Copyright: The Ohio State University
%         Date: 2011/12/06
%      Authors: Tom Zajdel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin == 0
    p = [];
end

p = Ensure_field(p, 'map_name', 'AIS');

p = Ensure_AIS_params(p);
p = Append_front_end_processes(p);
p = Append_process(p, 'FIR_filterbank_proc');
p = Append_process(p, 'HWR_proc');
p = Append_process(p, 'Integrate_fire_proc');
p = Append_process(p, 'LGF_proc_');
p = Append_process(p, 'Channel_mapping_proc_');
```

A.2 Inhibition_currents.m

```
function q = Inhibition_currents(t, A_inh, k, tau_inh)

% Inhibition_currents: Calculate inhibition currents for all ...
    channels at t
%
% function q = Inhibition_currents(alpha, base_level, sat_level)
%
% Inputs:
% t             - time from reference (typically uses the last spike ...
    time
%                 in each channel as a reference)
% A_inh         - maximum inhibition current amplitude
% k             - steepness rolloff factor
% tau_inh       - half value decay time constant
%
% Output:
% q             - the inhibition current values for all channels

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Copyright: Ohio State University
%         Date: 2011/12/22
%      Authors: Tom Zajdel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

q = A_inh ./ (1 + exp(k*(t-tau_inh)));
```

A.3 Integrate_fire_proc.m

```
function v = Integrate_fire_proc(p, u)

% Integrate_fire_proc: Race-to-spice algorithm.
% function v = Integrate_fire_proc(p, u)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Copyright: Ohio State University
%         Date: 2012/12/22
%      Authors: Tom Zajdel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch nargin

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 0  % Default parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    v = feval(mfilename, []);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 1  % Parameter calculations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Fundamental parameters:

    p = Ensure_field(p, 'A_inh',     0.5);
    p = Ensure_field(p, 'k',         1e4);
    p = Ensure_field(p, 'tau_avg', .3e-3);
    p = Ensure_field(p, 'tau_dev',     0);

    p = Ensure_field(p, 'threshold_avg', 100e-3);
    p = Ensure_field(p, 'threshold_dev',     0);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    v = p;  % Return parameters.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 2  % Processing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    v.magnitudes = zeros(size(u,2),1);
    v.channels = zeros(size(u,2),1);
    v.isis = zeros(size(u,2),1);                % interspike intervals
    n_prev = 1;
    index = 1;
```

```
47
48      % ensure initial conditions
49      channel_voltages = zeros(p.num_bands,1);
50      last_spike_times = -1e15*ones(p.num_bands,1);
51
52      % step through every audio sample, and race to integrate
53      for n=1:size(u,2)
54        t = n*p.timestep;
55        threshold = normrnd(p.threshold_avg,p.threshold_dev);
56
57        I_inh = Inhibition_currents(t - last_spike_times, p.A_inh, ...
58            p.k, normrnd(p.tau_avg,p.tau_dev) );
59
60        dv = max(u(:,n) - I_inh, 0);
61        channel_voltages = channel_voltages + dv;
62
63        % handle the spike if one channel has won
64        if max(channel_voltages) > threshold ...
65                && (n - n_prev)*p.timestep >= 1/p.implant_stim_rate
66          max_ch = max(channel_voltages) == channel_voltages;
67
68          channel_voltages(:)      = 0;  % reset voltages
69          last_spike_times(max_ch) = t;  % set last spike time for ...
               winning channel to penalize it in future
70
71          % store magnitude, firing channel, and ISI
72          v.magnitudes(index) = u(max_ch,n);
73          v.channels(index)   = find(max_ch);
74          v.isis(index)       = (n-n_prev)*p.timestep;
75
76          index  = index+1;
77          n_prev = n;
78        end
79      end
80
81      used         = 1:index-1;
82      v.magnitudes = v.magnitudes(used);
83      v.channels   = v.channels(used);
84      v.isis       = v.isis(used);
85
86  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
87  end
```

A.4 create_maps.m

```
function [ace,ais] = create_maps(c,t,client)
  % create_maps: sets map for experiment, adjusting for equal volume
  %
  % [ace,ais] = create_maps(c,t,client)
  %
  % c:      clinical map comfort levels
  % t:      clinical map threshold levels
  % client: client to stream to
  %
  % ace:    resulting ace map
  % ais:    resulting ais map

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %    Copyright: The Ohio State University
  %         Time: 2012/02/16
  %      Authors: Tom Zajdel
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % set up initial AIS current levels based on patient's ACE map
  ace = ACE_map;
  ais = AIS_map;

  ais.special_idle= 0;
  ace.phase_width = 25;
  ace.phase_gap    = 8;
  ace.special_idle = 0;
  ace.comfort_levels   = c(:);
  ace.threshold_levels = t(:);

  ais.threshold_avg = 5;
  ais.tau_avg = 0.1;

  % since pulse widths are shorter in AIS, we must adjust T and C levels
  % loop until the user says the two volumes are the same, adjusting ...
     DC every
  % iteration based on user feedback
  dc = 0;
  adjust = 1;
  while adjust ~= 0
    ais.threshold_levels = dc+ace.threshold_levels;
    ais.comfort_levels   = dc+ace.comfort_levels;

    test = create_session(1,25);

    run_session(test,ace,client,1);
    run_session(test,ais,client,1);
```

```
46
47      fprintf('\nDC offset = %d\n',dc);
48      adjust = input('Adjust by: ');
49      dc=dc+adjust;
50    end
51
52  end
```

A.5 main_experiment.m

```
% main_experiment: creates and runs experimental sessions

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Copyright: The Ohio State University
%         Date: 2012/02/16
%      Authors: Tom Zajdel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% select the appropriate map created during this MATLAB session ...
   using the
% create_maps function

map = ais; %or ace

f = [50 100 200 400]; % fundamental frequencies
test = create_session( 8,f); % training session
sess = create_session(40,f); % trial session

test_result = run_session(test,map,client); % get results
sess_result = run_session(sess,map,client);
```

A.6 create_session.m

```
function [s] = create_session(n_trials, freqs)
  % create_session: Create an experimental session
  %                 Randomly create a series of trials with an equal ...
     number
  %                 of trials for each frequency
  %
  % s = create_session(n_trials, freqs)
  %
  % n_trials: Total number of trials to run
  % freqs:    Vector of f0s to test
  %
  % s:        Random sequence of trials to run

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %    Copyright: The Ohio State University
  %         Date: 2012/02/16
  %      Authors: Tom Zajdel
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  s = [];
  s.freqs = freqs;

  % make sure each frequency gets the same number of trials
  s.trials = [];
  trials_per = floor(n_trials/length(freqs));
  n_trials   = trials_per*length(freqs);

  % create trials
  f = [];
  for k=1:length(freqs)
    f = [f; freqs(k)*ones(trials_per,1)];
  end
  for k=1:n_trials
    s.trials{k} = create_trial(f(k));
  end

  % randomize order
  s.trials = s.trials(randperm(length(s.trials)));
end
```

A.7 create_trial.m

```
function [t] = create_trial(f0)
  % create_trial: Given f0 initialize a random trial
  %               A trial consists of an Schr- pulse followed by a
  %               randomized Schr+ or Schr- pulse
  %
  % t = create_trial(f0)
  %
  % same:    Are the two pulses the same? 1=Same, 0=Diff
  % signals: Basis set of Schr+/- sequences at each f in the test
  % index:   Picks the f0 of the Schr+/- sequences in signals to use
  %
  % t:       Trial consisting of frequency, pulse sequence, and result

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %    Copyright: The Ohio State University
  %         Time: 2012/02/16
  %      Authors: Tom Zajdel
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  t = [];
  t.freq = f0;       % initialize frequency

  x = sign(randn);  % same determines the second pulse
  t.same = (x+1)/2; % same == 1 => [Schr- Schr-]
                    % same == 0 => [Schr- Schr+]

  t.success  = 0;   % success = 0 to start, will change to 1 during
                    % experiment if the user is correct
end
```

A.8 run_session.m

```
function [result] = run_session(s,map,client,vflag)
  % run_session: Runs an experimental section and gets results
  %
  % out = run_session(s, map, client)
  %
  % s:      Session to run
  % map:    MAP used to generate pulse sequences (ACE_map or AIS_map)
  % client: Client to stream to
  % vflag:  =1 if we are just checking volume
  %
  % result: Results matrix - % correct per frequency

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %    Copyright: The Ohio State University
  %         Date: 2012/02/14
  %      Authors: Tom Zajdel
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  if ~exist('vflag', 'var')  || isempty(vflag); vflag=0;  end

  % call the MAP to process all possible pulses and form basis sequences
  fprintf('\nWelcome to the Schroeder experiment! :)');
  fprintf('\nGenerating basis sequences...\n');
  n_pulses = 2*length(s.freqs);
  for k=1:2:n_pulses-1
    f=s.freqs((k-1)/2+1);
    signals{k}   = Process(map,schroeder(-f,0,400));
    fprintf('%3d Hz -Schr complete!\n',f);
    signals{k+1} = Process(map,schroeder( f,0,400));
    fprintf('%3d Hz +Schr complete!\n',f);
  end

  fprintf('Ready to begin! Press any key to start...\n');
  pause;

  % run through each trial in the sequence and set the success flag
  for k=1:length(s.trials)
    f   =s.trials{k}.freq;
    same=s.trials{k}.same;

    fprintf('\nTrial %d: %d Hz',k,f);

    if same
      cor='same';
    else
      cor='diff';
```

```
47      end
48
49      fprintf('\nThe correct answer is %s...\n',cor);
50
51      % create the sequence and ensure max period is 10e3
52      output = create_signal(same,signals,2*find(s.freqs==f)-1);
53      output.periods(output.periods > 10e3) = 10e3;
54
55      % create appropriate power_up sequence
56      power_up = output;
57      if length(output.periods)>1
58        power_up.periods = 100*ones(400,1);
59      end
60      power_up.electrodes = ones(400,1);
61      power_up.current_levels = zeros(400,1);
62
63      % Send the sequence to be streamed.
64      full_output = conc(power_up,output,10);
65      full_output = conc(full_output,power_up,10);
66      client = sendData(client, full_output);
67
68      % plot the stimulus
69  %     close all;
70  %     Plot_sequence_(full_output);
71  %     title(sprintf('%d Hz - %s',f,cor));
72
73      % Specify that the system is to wait for an external trigger.
74      % To start with an external trigger, the following command would ...
          be used:
75      % client = startType(client, TRIGGER.external);
76      TRIGGER = TriggerTypes;
77      client = startType(client, TRIGGER.immediate);
78
79      % Start the streaming
80      client = startStream(client);
81
82      % Wait until the streaming has finished.
83      pause(2);
84
85      % Stop the system
86      client = stopStream(client);
87
88  %     soundsc(schroeder_signal(f,same));
89  %     pause(2);
90
91      if ~vflag % skip the question if we are checking volume
92        s.trials{k}.success = input('Correct?  1=Yes  0=No\n');
93      end
94    end
95
96    result = s;
```

```
97
98    % return the result of the experiment for instant feedback!!
99    if ~vflag
100     r = analyze_session(s);
101     fprintf('\nThe experiment is finished!\nFreq\tCorrect\n');
102     for k=1:size(r,1)
103       fprintf('%d\t\t%6.2f%%\n',r(k,1),r(k,2)*100)
104     end
105     pause;
106   end
107
108 end
```

A.9 schroeder.m

```
function [v] = schroeder(f0,phi,t0,N,dB,fs)

% schroeder: Create and return schroeder phase signal with given
% fundamental frequency generated by an AUXMEX script
%
% v = schroeder(f0)
%
% f0: Fundamental frequency. If negative, construct negative ...
    schroeder, and
%     if positive, construct positive schroeder
% phi:Global phase, from 0 (0 rad) to 1 (2*pi rad)
% t0: Duration in miliseconds
% N:  Total number of harmonics
% dB: Relative sound level in dB
% fs: Sampling frequency in Hz
%
% v:  The resulting schroeder phase signal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Copyright: The Ohio State University
%         Date: 2012/01/31
%      Authors: Tom Zajdel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% set default parameters
if ~exist('t0', 'var')  || isempty(t0); t0=500;  end
if ~exist('N',  'var')  || isempty(N) ;  N =50;  end
if ~exist('dB', 'var')  || isempty(dB); dB=  5;  end
if ~exist('fs', 'var')  || isempty(fs); fs=16e3; end
if ~exist('phi', 'var') || isempty(phi);  phi=0; end

input = ...
    sprintf('ramp(sigma(i=1:%d,tone(i*%d,%d,%d+%d*(i*(i-1))/(%d*2))),100) ...
    @ %d', ...
        N,abs(f0),t0,phi,sign(f0),N,dB);

v = auxmex(struct('fs', fs),input);
```

A.10 plot_basis.m

```
function [] = plot_basis(freqs,map)
  % plot_basis: Plots the basis Schroeder sequences
  %
  % plot_basis(s, map)
  %
  % freq: fundamental frequencies to use
  % map:  MAP used to generate pulse sequences (ACE_map or AIS_map)

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %    Copyright: The Ohio State University
  %         Date: 2012/02/21
  %      Authors: Tom Zajdel
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  close all

  % generate basis sequences and plot them one frequency at a time
  fprintf('\nGenerating basis sequences...\n');
  n_pulses = 2*length(freqs);
  for k=1:2:n_pulses-1
    f=freqs((k-1)/2+1);
    signals{k}   = Process(map,schroeder(-f));
    fprintf('%3d Hz +Schr complete!\n',f);
    signals{k+1} = Process(map,schroeder( f));
    fprintf('%3d Hz -Schr complete!\n',f);

    % shift the plots to be side by side
    Plot_sequence_(signals{k});
    title(sprintf('%s Map: -Schr %d Hz',map.map_name,f));
    pos=get(gcf,'Position');    % move left by width/2
    pos(1)=pos(1)-pos(3)/2;
    set(gcf,'Position',pos);

    Plot_sequence_(signals{k+1});
    title(sprintf('%s Map: +Schr %d Hz',map.map_name,f));
    pos=get(gcf,'Position');    % move right by width/2
    pos(1)=pos(1)+pos(3)/2;
    set(gcf,'Position',pos);
    pause;
  end
```

A.11 create_signal.m

```
function [q]=create_signal(same,signals,index)
  % create_signal: Given a schroeder basis set and sequence, design ... signal
  %                Signal consists of a Shcr+ pulse followed by ... either a
  %                chr+ or Schr- pulse
  %
  % q = create_signal(same, signals, index)
  %
  % same:    Are the two pulses the same? 1=Same, 2=Diff
  % signals: Basis set of Schr+/- sequences at each f in the test
  % index:   Picks the f0 of the Schr+/- sequences in signals to use
  %
  % q:       Test sequence = [Schr- Schr+/-]

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %    Copyright: The Ohio State University
  %         Time: 2012/02/16
  %      Authors: Tom Zajdel
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  del = 500;              % msec delay between schroeder pulses
  q1  = signals{index};   % Schr- at f0 determined by index
  q2  = signals{index+1}; % Schr+ at f0 determined by index

  if same                 % pick either same or different condition
    q = conc(q1,q1,del);  % same: q = [Schr- Schr-]
  else
    q = conc(q1,q2,del);  % diff: q = [Schr- Schr+]
  end
end
```

A.12 conc.m

```
function [q] = conc(q1,q2,gapdur)
  % conc: Concatenate two pulse sequences separated by a gap.
  %
  % q = conc(q1, q2, gapdur)
  %
  % q1:     1st sequence struct
  % q2:     2nd sequence struct
  % gapdur: Duration of the gap in msec
  %
  % q:      Concatenated sequence = [q1 gap q2]

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %    Copyright: The Ohio State University
  %         Date: 2012/02/14
  %      Authors: Tom Zajdel, Bomjun Kwon
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Assume that phw and pg are the same for q1 and q2
  GapPeriod = .1;                              % use pulseperiod of ...
     100 usec
  nGapPulses = round(gapdur / GapPeriod);
  qgap.electrodes = ones(nGapPulses, 1);
  qgap.periods = GapPeriod * 1000 *ones(nGapPulses, 1);
  qgap.current_levels = zeros(nGapPulses, 1);

  q = q1;
  q.electrodes = [q.electrodes; qgap.electrodes; q2.electrodes];
  if length(q.periods)~=1  % only change the periods in the AIS case
    q.periods = [q.periods; qgap.periods; q2.periods];
  end

  q.current_levels = [q.current_levels; qgap.current_levels; ...
     q2.current_levels];
end
```

A.13 analyze_session.m

```
function [result] = analyze_session(s)
  % analyze_session: Analyze the success rate of a test session
  %
  % result = analyze_session(s)
  %
  % s: Completed session
  %
  % r: [frequency success rate] matrix

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %    Copyright: The Ohio State University
  %         Date: 2011/02/16
  %      Authors: Tom Zajdel
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  f = s.freqs;
  result = zeros(length(f),2);
  n_trials = length(s.trials);
  n_freqs  = length(f);

  % step through each of the frequencies and calculate the success rate
  for m=1:n_freqs
    result(m,1)=f(m);
    k = 0;
    guess = zeros(1,n_trials/n_freqs);

    if length(guess)==1         % handle special 1-trial case
       result(1,2) = guess;
       return;
    end

    for n=1:n_trials
      if s.trials{n}.freq == f(m);
        k=k+1;
        guess(k) = s.trials{n}.success;
      end

    end

    result(m,2)= sum(guess)/length(guess);
  end

end
```

BIBLIOGRAPHY

[1] J. Sit, A. M. Simonson, A. J. Oxenham, M. A. Faltys, and R. Sarpeshkar, "A low-power asynchronous interleaved sampling algorithm for cochlear implants that encodes envelope and phase information," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 1, pp. 138–149, 2007.

[2] S. Peng, N. Lu, and M. Chatterjee, "Effects of cooperating and conflicting cues on speech intonation recognition by cochlear implant users and normal hearing listeners," *Audiology and Neurotology*, vol. 14, no. 5, pp. 327–337, 2009.

[3] M. Chatterjee and S. Peng, "Processing F0 with cochlear implants: modulation frequency discrimination and speech intonation recognition," *Hearing Research*, vol. 235, no. 1-2, pp. 143–156, 2008.

[4] S. Peng, J. B. Tomblin, and C. W. Turner, "Production and perception of speech intonation in pediatric cochlear implant recipients and individuals with normal hearing," *Ear and Hearing*, vol. 29, no. 3, pp. 336–351, 2008.

[5] K. K. Paliwal, "Usefulness of phase in speech processing," in *Proceedings of IPSJ Spoken Language Processing Workshop*, 2003, pp. 1–6.

[6] W. A. Yost, *Fundamentals of Hearing: An Introduction.* San Diego: Academic Press, 2006.

[7] Boston University Neuromorphics Laboratory, "Auditory system," 2011, accessed 22-Apr-2012. [Online]. Available: http://nl.bu.edu/research/projects/moneta/moneta-v2-0/auditory-system/

[8] A. J. Hudspeth, "Making an effort to listen: mechanical amplification in the ear," *Neuron*, vol. 59, no. 4, pp. 530–545, 2008.

[9] Wikipedia, "Cochlear implant," 2012, accessed 22-Apr-2012. [Online]. Available: http://en.wikipedia.org/wiki/Cochlear_implant

[10] "ACE and CIS DSP Strategies: Software Requirements Specification," N95287F Issue 1. Cochlear Corporation, New South Wales, Australia, October 2002.

[11] A. Alvarez, R. D'sa, M. Toure, and T. Zajdel, "Social Enterprise and Cochlear Implants." Final report for Ohio State University BUSMHR 691 course project, Autumn 2011.

[12] A. Krenmayr, "Christian Doppler Laboratory for Active Implantable Systems," 2010, accessed 07-May-2012. [Online]. Available: http://www.uibk.ac.at/activeimplants/home_en.htm

[13] "Nucleus MATLAB Toolbox 2.11: Software User Manual," N95246F Issue 1. Cochlear Corporation, New South Wales, Australia, October 2002.

[14] B. J. Kwon, "AUX: A scripting language for auditory signal processing and software packages for psychoacoustic experiments and education," 2011, in press.

[15] G. Ohm, "Uber die Definition des Tones, nebst daran geknupfter Theorie der Sirene und ahnlicher tonbildender Vorrichtungen," *Annalen der Physik und Chemie*, vol. 59, p. 513565, 1843.

[16] M. R. Schroeder, "Synthesis of low-peak-factor signals and binary sequences with low autocorrelation," *IEEE Transactions on Information Theory*, vol. 16, no. 1, pp. 85–89, 1970.

[17] W. R. Drennan, J. K. Longnion, C. Ruffin, and J. T. Rubinstein, "Discrimination of schroeder-phase harmonic complexes by normal-hearing and cochlear-implant listeners," *Journal of the Association for Research in Otolaryngology*, vol. 9, no. 1, pp. 138–149, 2008.

[18] W. R. Drennan, J. H. Won, K. Nie, E. Jameyson, and J. T. Rubinstein, "Sensitivity of psychophysical measures to signal processor modifications in cochlear implant users," *Hearing Research*, vol. 262, no. 1, pp. 1–8, 2010.